COTS Software Supplier Identification and Evaluation

April 2000

Ann Miller
Cynthia Tang Missouri Distinguished Professor of Computer Engineering
Department of Electrical and Computer Engineering
University of Missouri – Rolla
1870 Miner Circle
Rolla, MO 65409 USA

Abstract. There has been a consistent trend to field increasingly large systems. Largeness requires a longer development cycle that is in direct conflict with the need to field systems quickly. Several approaches have been developed to reduce time-to-market. One of the most notable methods in reaction to time-to-field pressures is the inclusion of Commercial-Off-the-Shelf (COTS) as well as Government-off-the-Shelf (GOTS) software packages to perform some of the functions of these new "mega-systems". This paper addresses some of the advantages and pitfalls of the inclusion of COTS components and discusses the need for an evaluation not only of the COTS component but also of the COTS supplier. The paper concludes with some of the lessons learned from the use of COTS incorporation and of supplier assessments over a ten-year span of commercial and government acquisitions.

Keywords. Large project development, COTS components, software acquisition strategy, software supplier evaluation.

Introduction. Large-scale systems are becoming increasingly common, both in military and commercial systems. As systems provide more features and functions, the size of the delivered software increases as well. Sheer size, whether measured in thousands of lines of code (KLOC) or in bytes of program code, is one metric by which to gauge the "largeness" of a system. Measured by size, software content in systems seems to be following a software variant of Moore's Law [1] with exponential increases in size every generation, or approximately every 18 months if the systems are not related by product line. Another indicator of growth is development team size. Complexity and function point metrics are other possible indicators. Some projects have also used pages of documentation as a metric; the author recalls one project in the mid-1980s for which the requirements specification documents in ring-bound notebooks spanned more than six linear feet of shelf space. Whatever measure is chosen for the yardstick, numerous examples of large-scale systems can be found.

As with any task, scale has its effect on software and systems development. An individual can assemble an ultralight plane from a kit; so, too, can one individual design, code, and test a small software program. But the ultralight builder cannot undertake the sole design, development, manufacture, and assembly of a Boeing 777 aircraft. Neither can one software engineer undertake the sole design, development, code, and test of a large-scale software program. In addition to the sheer length of time for such an undertaking, fielding of a large program requires a multitude of skills, as does the assembly of the 777 jet. Thus, a large team is necessary.

Typically, no single organization has all of the expertise to bring a large-scale product to market; even if the expertise were present, it might be unrealistic to apply all of the organization's resources to one product. Thus, development of a large system will likely include suppliers for portions of the hardware and/or software. In addition, there is typically a geographic disbursement not only between the development organization and the software

Paper presented at the RTO IST Symposium on "Commercial Off-the-Shelf Products in Defence Applications "The Ruthless Pursuit of COTS"", held in Brussels, Belgium, 3-5 April 2000, and published in RTO MP-48.

suppliers, but also among the various development teams.

This paper will discuss incorporation of COTS components into large-scale development efforts and will provide some lessons learned from more than a decade of technical contribution on and management oversight of large programs.

Why is this topic important? Capers Jones [2] has summarized it most succinctly: "Software package acquisition actually delivers more software to business and government users than almost any kind of development activity. Yet in spite of the huge volumes of software purchased or leased every year by companies, civilian government agencies, and military services, the process of acquiring packages is curiously amateurish and unprofessional. Some organizations have no formal methodologies for package evaluations and acquisition."

This paper describes a formal software supplier identification and evaluation process that began in 1990 and that has evolved over the years. Specifically, we will examine the need for a structured evaluation of the COTS vendor as well as an evaluation of the COTS product itself. This is not a research paper; it is presented to practitioners by a practitioner. No theorems will be proven, no formal assertions will be derived, no names will be named. However, examples from commercial and government acquisitions will be discussed and references will be provided which span a large spectrum devoted to the topic of including COTS packages in large-scale software development.

The Effects of Scale and Time-to-Market. A large team brings its own set of problems, and the effects of scale on a project have been discussed in the literature, from communication nodes to function points to general project management. One of the most insightful articles concerning scale discussed organizational and management aspects of large projects in terms of the Tower of Babel [3]. The criticality of architecture [4] and testing [5] of large systems has also been analyzed from the practitioner's viewpoint. Life-cycle models for the large systems have been discussed and have evolved over time. [6, 7, 8, 9]. Lastly, there have been numerous papers devoted to development processes for large-scale systems, with many of these based on the Software Engineering Institute's (SEI) Software Capability Maturity Model [10]. Lastly, both commercial [11] and military software acquisition standards have emerged for summarizing best practices. But no matter which life-cycle model and development process and methodology adopted, the basic effect of largeness on a product is that it takes longer to build.

On the other hand, in industry, time-to-market considerations foster rapid fielding of systems, exactly opposite to the effect of scale on a project. Military applications have similar pressures. Once new technology is available to the warfighter, there is the strong desire to ruggedize the hardware component and distribute that technology. Both commercial development organization and military acquisition offices have sought ways out of this conflicting situation. A common approach to field large systems in a timely manner is inclusion of COTS packages to provide portions of the total system functionality. Large-scale government systems may contain GOTS packages as well.

Reasons for COTS Components. By incorporating COTS components in a larger system, the development time for that functionality is decreased; however, such a practice is not a panacea. Just because a portion of coding has been eliminated, the remaining phases of analysis, design, integration testing and acceptance testing remain. Still, there are many reasons to consider COTS packages in addition to the savings on development time: (i) the particular packages might require a specific domain expertise which does not reside in the development organization, (ii) the package may be a de-facto standard which customers expect to be part of the total system, or (iii) it might be an existing product from another part of the development organization which is being reused as part of a business plan to enter new markets.

The Software Acquisition Strategy. The first issue related to any large-scale effort is to determine the acquisition strategy. Acquisition examines three distinct questions. First, what portions of the total system already exist, either through re-used software or through a COTS component? Second, which portions must be developed? Lastly, of those portions that must be developed, which can be subcontracted and which should be developed in-house? In-house development is often reserved for the "family

jewels", that is, those portions of the final product that would give the developing organization a competitive edge in the marketplace.

In 1990, the author was the Chief Software Engineer for a large commercial project in which the software acquisition priority was to seek COTS components for as much as possible beyond those aspects which had been identified as required for internal development due to competitive advantage. If COTS packages could not be found, then and only then, would we seek qualified suppliers to generate the remaining functionality. In typical systems development projects, an enterprise will develop most of the product themselves. However, for this product, the initial software size estimate was 12 Million Lines of Code (MLOC) and the critical proprietary code was estimated to form approximately 8% of that total. Thus, with more than 10 MLOC to be subcontracted, it made good business sense to generate a software supplier identification and evaluation plan.

Identification of COTS Components. Once the decision to procure a component has been made, a careful market analysis of potential packages must be made. There are several concerns related to identification of such software packages. The first issue is functionality. A COTS package will most likely not be an exact fit; that is, it may not have all of the required features or it may have additional, unwanted features relative to the system requirements. Since the large-scale system developer probably will not have access to the source code of the COTS package, can the developer assure that the package performs its intended functions and that unwanted features won't be able to be invoked when integrated into the whole? Compatibility is another issue; the COTS package itself will most likely be evolving. What is the vendor's release plan? Typically, a customer incorporating a COTS component does not have control or influence in the package's evolution. Will future releases be backward compatible? What is the package's quality and reliability? If the package is plagued with numerous patches between scheduled releases, testing time for the larger system increases. Large systems tend to be long lived once they are fielded due to the significant investment in development. Therefore, the quality, reliability and trustworthiness of the COTS package are critical considerations. If the package includes features that are not going to be implemented in the larger system, can feature blocking be assured or will these functions be a potential cause for total system failure or degradation of service? Military systems are prime targets for hackers; many times hackers find their way into a government system by defects in COTS components. Another concern is obsolescence. Will the package become outdated by the time it is fielded in the larger system? This is a consideration because of the long development time of the larger system and because that such systems typically have a long half-life.

Evaluation of COTS Components. Once a set of potential packages are identified, the next step is evaluation of the contenders. The product evaluation should include quality and reliability as well as functionality and performance. In addition to the "black-box" evaluation of the product, the requirement for the product's functionality and its interface within the total system should be carefully defined and documented. The evaluation process should down-select the candidate packages to a small number. In some cases, our initial search for products located up to 100 potential packages and exhaustive investigation would reduce the number to a short list of 10 or fewer. More detailed product tests would then be performed on those packages.

Supplier Identification and Evaluation Process. Once the short list of products have been evaluated and some potentially eliminated from consideration, it is time for an assessment of the vendors of those remaining products. The initial phase of the supplier evaluation process should examine the package's overall score in the product evaluation, which include not only performance and reliability but other factors such as cost and other consumer evaluations. We typically would determine the top three contenders and proceed to the second phase with this set. This next phase would consider three factors: (i) the domain expertise of the vendor, (ii) the business and financial health of the vendor, and (iii) the results of an on-site process assessment of the vendor.

We felt that the selection of a COTS software supplier should be a variation of the selection

process for any software supplier [12]. So the primary, but not sole, factor is the technical expertise of the vendor. In addition to technical concerns, there are business issues to consider. Is the supplier's company financially sound? Even if the source code is held in escrow, it may not be easily supported if the vendor has gone out of business. And even if the vendor stays in business, will the software continue to be supported? Customer service and responsiveness are factors to consider. Thus, we felt that an on-site visit would be required. Each of the business/financial analysis results and the supplier process assessment results served as GO/NO GO decision gates.

Software Supplier Assessment. The evaluation of a vendor should include not only the quality and reliability of the product but also the quality of the vendor's configuration management and release processes. These assessments do not need to be long, arduous evaluations. They can be streamlined to fit the size and scope of both the product and the vendor. Basically, the question is: What is the real cost of the software package? The total costs include not only the licensing, the integration and interface testing in the larger system, but also training, long-term maintenance, and the management of upgrades over time. The answers to these cannot be directly calculated but can be indirectly approximated by a combination of product evaluation, business analysis, and supplier assessment.

Two entities can greatly assist in the identification and evaluation of COTS components: the requirements specifications and the interface control document (ICD). The former helps to answer questions related to the applicability of the package; the latter helps to scope the level of effort needed to incorporate the package, thereby determining some of the hidden costs. The ICD is also a critical factor in testing of a large system with integrated COTS components. With a well-written and structured ICD, re-usable interface tests can be developed. The ability to reuse and/or automate tests becomes crucial. Incremental test planning is a necessity in any large-scale system because the system is evolving. Whenever COTS packages are incorporated, additional tests concerning error handling at the interface should be designed. In addition, performance and stress testing of the component's interface should be conducted. The buy versus make decision should be based on a realistic estimate of the total cost of each effort.

Structure of On-Site Visit. Because of the magnitude of software being contracted in the first system mentioned, we developed a supplier process assessment that could be tailored for the size of the company and for the type of software activity. If the potential supplier would be developing software, the visit ranged from one to three days depending on the size of organization. For COTS products, the visit was streamlined to a one-day visit regardless of the size of the company.

We structured the supplier process assessment using the SEI Capability Maturity Model as the basis. The vendor evaluation had many similarities to an SEI Software Process Assessment (SPA). Both use an interview and discussion format and we selected many of our questions from the SEI questionnaire. In addition, the lead of the evaluation team was always a trained SEI assessor. The evaluation teams generally were had a total of three members, which is a reduction from the SEI assessment team size. The development team which would be incorporating the package was always represented on the assessment team. The third member would come from any part of the development team or from the contracts organization which had professionals who specialized in software contracting. If these non-lead members were not certified SEI assessors, then they received a short in-house training course in the supplier evaluation process prior to visiting the vendor. One of the differences from the initial SEI assessment format is that we scheduled private interview sessions with a senior executive manager and with the chief technical officer or chief scientist or chief software engineer (depending on the vendor's organization). For the COTS vendors, we also spent more time with the Quality Assurance team and with the configuration management team than with the development and test teams. We also met with customer service and support teams of the potential COTS supplier. Another difference is that the SEI questionnaire, while a basis for the discussions, was not completed by the vendor in advance. Lastly, as with an SEI assessment, we did present a findings session at the close of the visit. The findings presentation included a supplier rating, but did not determine an SEI maturity level from 1 through 5. Rather,

the rating was one of fully qualified, qualified, or not qualified. Most of the suppliers ranked in that middle category. However, in the most important portion of the findings, we listed what we perceived to be the strengths and weaknesses of the supplier. Improvement in weak areas became contract requirements if we chose to pursue the relationship further.

In a commercial satellite communications example, several packages related to orbital analysis and telemetry tracking and control that were efficiently incorporated due in large part to the careful supplier evaluation. In another case, the board of directors of a small company mandated the company's president to address the action items resulting from our supplier evaluation regardless of whether we entered into a contract. On the flip side, we terminated a contract with one supplier who had excellent domain expertise because the first deliverables from the organization were extremely poor; the root causes of the poor quality were the very areas identified as opportunities to improve from the supplier evaluation.

Assessment Follow-On. We instituted a mechanism of communications and follow-on with our contracted suppliers. This included a management forum of quarterly meetings of senior executives, primarily from those suppliers who were developing software. There was also a periodic technical forum that included all of our suppliers. This was especially important in the use of one COTS product, namely the mandated configuration management tool. We kept open the option of re-evaluation. These re-evaluations were based on contractually required improvements (if any), and issues of concern from the sub-contract managers. The follow-on visits were informal with mutual presentations, questions, and discussions. With all of the suppliers, we provided an opportunity to evaluate us and let us know how well we were doing as contract managers. We felt that this is an important aspect of growing a long-term supplier relationship. Total improvements can best be made in a spirit of constructive and honest feedback.

Conclusions and Lessons Learned. The following are some of the findings related to implementing supplier process and product evaluations in large-scale development, which include a mix of both commercial and government systems.

First, the use of COTS packages can reduce total system development time, but the savings are partially offset by increased design time up front and increased interface testing downstream. It is critical to scope out the total life-cycle cost of the COTS package, not merely the licensing cost. A formal COTS identification and evaluation process can greatly assist in this scoping effort. The COTS evaluation should include an assessment of the vendor's process as well as of the package itself. The decision to incorporate the COTS component should be based on product, process, and business factors.

Lastly, a hard lesson learned concerns whether or not to modify a COTS package. Since the package is most likely not a perfect fit, there is a tendency to work with the vendor and modify the package. The short version of the lesson is: DON'T. If you feel that you must modify the package, modify your process first. Then, and only then, if you absolutely must modify the package, build a wrapper and still do not modify the package. If that still doesn't convince you, perhaps simple economics will. If you modify the COTS package, you may void the warranty. If you contract with the vendor to modify the package, you will be contracting with them for the life of your product.

In summary, large-scale systems are difficult project development activities with significant time constraints and with anticipated but unknown changes. Inclusion of COTS components can reduce total life cycle costs and support the successful fielding of a quality product. The quality and reliability of that system can be improved with a COTS product and process evaluation.

References.

1. Tom DeMarco and Ann Miller, "Managing Large Software Projects", *IEEE Software*, July 1996.
2. Frederick P. Brooks, Jr., *The Mythical Man Month, Essays on Software Engineering*, Addison-Wesley, 1975.
3. Capers Jones, *Patterns of Software Systems Failure and Success*, International Thomson Computer Press, 1996.

4. Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice*, Addison-Wesley, 1998.
5. Daniel M. Marks, *Testing Very Big Systems*, McGraw-Hill, 1992.
6. W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques", originally published in *Proceedings of WESCON*, August 1970; also available in *Proceedings of 9th International Conference on Software Engineering (ICSE 9)*, IEEE/ACM, 1987.
7. Bill Curtis, Herb Krasner, Vincent Shen, and Neil Iscoe, "On Building Software Process Models Under the Lamppost", *Proceedings of 9th International Conference on Software Engineering (ICSE 9)*, IEEE/ACM, 1987.
8. Barry Bochm, "Anchoring the Software Process", *IEEE Software*, July 1996.
9. Ann Miller, "Design and Test of Large-Scale Systems", *Joint Proceedings of the International Conference on Software Management and International Conference on Applications of Software Measurement*, March 2000.
10. Watts Humphrey et al, *A Method for Assessing the Software Engineering Capability of Contractors*, Technical Report CMU/SEI-87-TR23, Software Engineering Institute, 1987.
11. IEEE Standard-1062, *Recommended practice for Software Acquisition*, Institute of Electrical and Electronics Engineers, 1993.
12. Jim Nielsen and Ann Miller, "Selecting Software Suppliers", *IEEE Software*, July 1996.